Bevezetés a Python programozási nyelvbe

Szathmáry László
Debreceni Egyetem
Informatikai Kar

1. Gyakorlat (2012. szept. 20.)
- bevezető
- a sztring adattípus

A tantárgyról

A tantárgy neve: Bevezetés a Python programozási nyelvbe

A tantárgy kódja: IN[BCD]V381L

A tárgy BSc-s hallgatók számára lett meghirdetve (PTI, MI, GI).

Előfeltétel: Magas szintű programozási nyelvek 1

A gyakorlat honlapja:

http://www.inf.unideb.hu/~szathml/pmwiki/index.php?n=Acad.Py

A gyakorlatok ideje és helye:

- csütörtök 8h-10h, IK-105
- csütörtök 10h-12h, IK-105

Követelmények

Az aláírás megszerzésének **egyik** feltétele a rendszeres részvétel a gyakorlatokon. A félév során legfeljebb 3 hiányzás megengedett. Aki ezt túllépi, annak az aláírás automatikusan megtagadásra kerül.

A szorgalmi időszakban egy 50 perces közös **zárthelyi dolgozat** megírására kerül sor. Ennek időpontja és helyszíne:

- 2012. november 14., szerda, IK-F0, 18h-tól

Vizsgára csak az jöhet, aki legalább 50%-osra teljesíti a ZH-t. Sikertelen ZH esetén van javítási lehetőség a vizsgaidőszak első hetében.

A vizsga **szóban** történik számítógép mellett. A hallgatónak **néhány konkrét programozási feladatot** kell helyben megoldania, illetve ismernie kell a Python programozási nyelvvel kapcsolatos fogalmakat.

Ajánlott irodalom

- *Guido van Rossum*: **Python Tutorial** (http://docs.python.org/tutorial/index.html, PDF)
- *Wesley J. Chun*: **Core Python Programming** (2nd Edition)
- *Doug Hellmann*: **The Python Standard Library by Example** (Developer's Library)
  [online változat: **Python Module of the Week** (http://www.doughellmann.com/PyMOTW/contents.html)]
- *Zed A. Shaw*: **Learn Python The Hard Way** (http://learnpythonthehardway.org/)
- *Allen B. Downey*: **Think Python** (How to Think Like a Computer Scientist) http://www.greenteapress.com/thinkpython/
- *Paul Barry*: **Head First Python**
- *Mark Pilgrim*: **Dive Into Python 3** (http://www.diveintopython3.net/ , ez már kimondottan a Python 3-ról szól)

Ajánlott irodalom (folyt.)

- *Gérard Swinnen:* **Tanuljunk meg programozni Python nyelven**
  (2005, Python 2.2)
  (online letölthető: http://python.free-h.net/spip.php?article4)
- *Rashi Gupta:* **Mindentudó Python**
  (2003, Python 2.2)

Bevezető

- A Python egy általános célú, nagyon magas szintű programozási nyelv.
- Fő tervezési szempont: olvashatóság.
- Interpreteres nyelv, a megírt program azonnal futtatható.
- Multiparadigmás (imperatív, objektumorientált, funkcionális).
- Az első változat 1991-ben jelent meg.
  Tervezője Guido van Rossum holland kutató/programozó (2005 óta a Google-nél dolgozik).
  Nevét a Monty Python csoportról kapta.
- Mely nyelvek voltak rá hatással: ABC, ALGOL 68, C, C++, Dylan, Haskell, Icon, Java, Lisp, Modula-3, Perl.
- Mely nyelvekre volt hatással: Boo, Cobra, D, Falcon, Groovy, JavaScript, Ruby.

Bevezető

- Dinamikus típusokat és automatikus memóriakezelést használ.
- Platformfüggetlen (Unix/Linux, Windows, Mac OS, stb.)
- A Pythonnak igen kiterjedt és széles körű standard könyvtára van („batteries included”), amit még kiegészítenek az egyéb (mások által megírt) publikus modulok („3rd party modules”).
- Az interpreter és a standard könyvtár teljesen nyílt forrású.
- Könnyen tanulható, egyszerű a szintaxisa. A megírt kód könnyen olvasható.
- A programozói munkát hatékony magas szintű adatszerkezetek segítik. Egyszerűen, ugyanakkor nagyon hatásosan valósítja meg az objektumorientált programozást.

Bevezető

- Ideális nyelv szkriptek írásához, illetve gyors alkalmazásfejlesztéshez („rapid application development”).
- Gyors prototípusfejlesztést tesz lehetővé („rapid prototyping”).
- Hasonló programozási nyelvek: Perl, Ruby.
- Tökéletes választás kisebb (pl. 10-20 soros) szkriptekhez, de NEM CSAK erre jó! Nagy méretű, több ezer soros programokat is lehet benne írni úgy, hogy a program áttekinthető marad (modulok, csomagok).
- Két ág létezik: Python 2.7.x és Python 3. A 2.7-es széria kiforrott, széles körben támogatott. A jelenlegi és jövőbeli fejlesztések a 3-as szériára koncentrálnak. A gyakorlaton a Python 2.7-es verziót fogjuk használni.

Linkek

- Python HQ: http://www.python.org/
- Python dokumentáció: http://docs.python.org/
- A Python Standard Library: http://docs.python.org/library/
- Python FAQ: http://docs.python.org/faq/general.html
- PEP 8 -- Style Guide for Python Code: http://www.python.org/dev/peps/pep-0008/
- Python 2 vagy Python 3? http://wiki.python.org/moin/Python2orPython3

- http://www.reddit.com/r/learnpython
- http://www.reddit.com/r/python
- http://stackoverflow.com/questions/tagged/python

Hol használják?

- Python sikertörténetek: http://www.python.org/about/success/

- Scientific
  - Biology
  - Bioinformatics
  - Computational Chemistry
  - Data Visualization
  - Drug Discovery
  - GIS and Mapping
  - Scientific Programing
  - Simulation
  - Weather

Mars Curiosity (2012. aug. 6.)

Szoftver: 2,5 millió C programsor.

A logfile-ok tesztelését Python szkriptekkel végezték.

- Google (C++ / Python / Java)
  „Python where we can,
  C++ where we must”
  (link)

Mennyire népszerű?

TIOBE index (http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html)

Position Jul 2012	Position Jul 2011	Delta in Position	Programming Language	Ratings Jul 2012	Delta Jul 2011	Status
1	2	↑	C	18.331%	+1.05%	A
2	1	↓	Java	16.087%	-3.16%	A
3	6	↑↑↑	Objective-C	9.335%	+4.15%	A
4	3	↓	C++	9.118%	+0.10%	A
5	4	↓	C#	6.668%	+0.45%	A
6	7	↑	(Visual) Basic	5.695%	+0.59%	A
7	5	↓↓	PHP	5.012%	-1.17%	A
8	8	=	Python	4.000%	+0.42%	A
9	9	=	Perl	2.053%	-0.28%	A
10	12	↑↑	Ruby	1.768%	+0.44%	A
11	10	↓	JavaScript	1.454%	-0.79%	A
12	14	↑↑	Delphi/Object Pascal	1.157%	+0.27%	A
13	13	=	Lisp	0.997%	+0.09%	A
14	15	↑	Transact-SQL	0.954%	+0.15%	A
15	25	↑↑↑↑↑↑↑↑↑↑	Visual Basic .NET	0.917%	+0.43%	A
16	16	=	Pascal	0.837%	+0.17%	A
17	19	↑↑	Ada	0.689%	+0.14%	B
18	11	↓↓↓↓↓↓↓	Lua	0.684%	-0.89%	B
19	21	↑↑	PL/SQL	0.645%	+0.10%	A--
20	26	↑↑↑↑↑↑	MATLAB	0.639%	+0.19%	B

Munkalehetőségek: http://careers.stackoverflow.com/jobs?searchTerm=python

Interpreter használata:

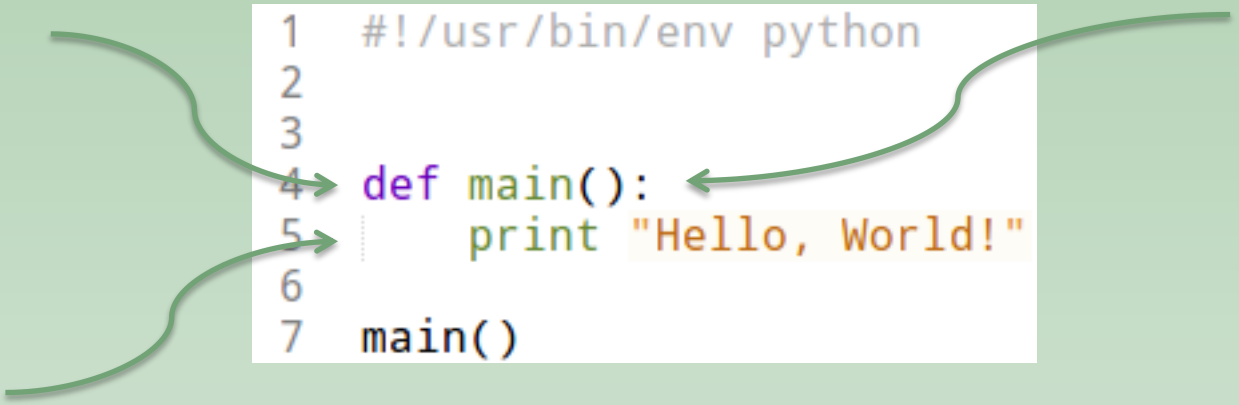
```
1  Python 2.7.3 (default, Aug  1 2012, 05:14:39)
2  [GCC 4.6.3] on linux2
3  Type "help", "copyright", "credits" or "license" for more information.
4  >>> "Hello, World!"
5  'Hello, World!'
```

Szkript írása:

```
1  #!/usr/bin/env python
2
3  print "Hello, World!"
```

```
>>> a = 6
>>> a
6
>>> a = "hello"
>>> len(a)
5
>>> a
'hello'
>>> A
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'A' is not defined
>>> "hello " + "world"
'hello world'
>>> "hello " + 6
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> "hello " + str(6)
'hello 6'
```

változót nem kell külön deklarálni



```
1  #!/usr/bin/env python
2
3
4  def main():
5      print "Hello, World!"
6
7  main()
```

```
1  #!/usr/bin/env python
2
3
4  def main():
5      print "Hello, World!"
6
7  if __name__ == "__main__":
8      main()
```

Direkt módon futtatjuk vagy modulként?

Írassuk ki a parancssori argumentumokat:

```
 3  import sys
 4
 5
 6  def main():
 7      print sys.argv
 8
 9  if __name__ == "__main__":
10      main()
```

A továbbiakban nem írjuk ki külön a `#!/usr/bin/env python` sort…

Majd: adjunk meg egy nevet argumentumként (pl. `./hello.py Bob`), s üdvözöljük az illetőt („`Hello Bob !`”).

```
 3 import sys
 4
 5
 6 def hello(name):
 7     if name == 'Batman' or name == 'Robin':
 8         print 'Batman vagy Robin'
 9     else:
10         print NincsIlyenFuggveny()
11
12
13 def main():
14     hello(sys.argv[1])
15
16 if __name__ == "__main__":
17     main()
```

Csak akkor derül ki a hiba, ha idekerül a vezérlés!

Ezért (is) lényegesek a unit tesztek komolyabb rendszerek esetén.
Minden ágat le kell velük tesztelni.

Sztringek

```
>>> s = "Hello"
>>> s
'Hello'
>>> s = 'Hello'
>>> s
'Hello'
>>> s = "isn't"
>>> s
"isn't"
>>> s = 'he said: "go home"'
>>> s
'he said: "go home"'
>>> s = "he said: \"go home\""
>>> s
'he said: "go home"'
>>> s = 'batman'
>>> len(s)
6
>>> s[0]
'b'
>>> s[0] = 'B'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> s
'batman'
>>> s + '!'
'batman!'
>>> s = 'Joker'
>>> s.lower()
'joker'
>>> s.upper()
'JOKER'
>>> s.find('k')
2
>>> s.find('a')
-1
>>> s[20]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

Sztring metódusok:
http://docs.python.org/library/stdtypes.html#string-methods

strings are *immutable* objects (read-only)

HF: válasszunk ki egy sztring metódust s írjunk egy kis szkriptet ami bemutatja a használatát.

Néhány gyakori sztring metódus

`s.lower(), s.upper()`
a sztring kisbetűs, nagybetűs verziójával tér vissza

`s.strip()`
a whitespace karaktereket levágja a sztring elejéről és végéről

`s.isalpha()` / `s.isdigit()` / `s.isspace()`...
megnézi, hogy a sztring vmennyi karaktere az adott karakterosztályba tartozik-e

`s.startswith('other'), s.endswith('other')`
megnézi, hogy a sztring a másik sztringgel kezdődik-e / végződik-e

`s.find('other')`
A sztringben szerepel-e a másik sztring (nem reguláris kifejezésként adjuk meg). Ha igen, akkor az első előfordulás első karakterének indexével tér vissza. Ha nem, akkor -1 a visszatérési érték.

`s.replace('old', 'new')`
a sztringben az 'old' vmennyi előfordulását 'new'-ra cseréli

`s.split('delim')`
A sztringet az adott szeparátor mentén részsztringek listájára bontja. A szeparátor nem reguláris kifejezés. Példa: 'aaa,bbb,ccc'.split(',') -> ['aaa', 'bbb', 'ccc']. Ha csak s.split() -et írunk, akkor a whitespace karakterek mentén bontja fel a sztringet.

`s.join(list)`
A split() ellentéte. Egy lista elemeit kapcsolja össze egy adott szeparátorral (ez lesz az s sztring). Példa: '---'.join(['aaa', 'bbb', 'ccc']) -> aaa---bbb---ccc.